

Controle de Concorrência

GABRIELLI DANKER
MARCO A. SAMUELSSON





Conceito de Controle de Concorrência

É um conjunto de técnicas e mecanismos projetados para garantir que múltiplas transações concorrentes possam ser executadas de maneira eficiente, sem interferências indevidas, mantendo a consistência dos dados.

Em ambientes de banco de dados onde várias transações podem ocorrer simultaneamente, o controle de concorrência é essencial para evitar problemas como leituras sujas, escritas sujas, leituras fantasmas e outros fenômenos indesejados.



Problemas



LEITURA SUJA (DIRTY READ)

Ocorre quando uma transação lê dados que foram modificados por outra transação, mas essa segunda transação ainda não foi confirmada (não fez commit). Se a segunda transação for posteriormente desfeita (rollback), a leitura inicial terá sido baseada em dados que nunca deveriam ter sido visíveis.

ESCRITA SUJA (DIRTY WRITE)

Ocorre quando uma transação modifica dados que foram lidos por outra transação, mas essa primeira transação ainda não foi confirmada. Se a primeira transação for desfeita, a segunda transação terá modificado dados que não deveriam ter sido alterados.

LEITURA FANTASMA (PHANTOM READ)

Acontece quando uma transação executa uma consulta que retorna um conjunto de registros, mas outra transação insere ou remove registros que afetariam o resultado da consulta. A transação que executa a leitura pode ver resultados que não existiam quando iniciou.

ESCRITA FANTASMA (PHANTOM WRITE)

Ocorre quando uma transação insere ou remove registros que afetam o resultado de uma consulta executada por outra transação. A transação que executa a consulta pode perceber mudanças que não existiam quando iniciou.

Técnicas de Controle



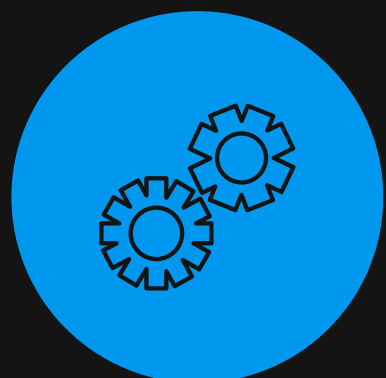
Bloqueio (Locking)



Controle de Versão
(Versioning)



Isolamento de Snapshot
(Snapshot Isolation)



Baseado em Tempo
(Timestamp-based)

Bloqueio (Locking)

Essa técnica envolve a aquisição de bloqueios exclusivos por transações ao acessar ou modificar dados específicos. Enquanto o bloqueio estiver ativo, nenhum outro usuário pode modificar o mesmo dado, assegurando que apenas uma transação por vez o altere e prevenindo inconsistências. Diversos tipos de bloqueios, incluindo Bloqueio Binário, Bloqueio Múltiplo e Bloqueio em Duas Fases, são utilizados para esse fim.



Bloqueio Binário

Um bloqueio binário possui dois estados:

- bloqueado (locked);
- desbloqueado (unlocked).

As operações necessárias são:

- lock_item(X): bloqueia o item X;
- unlock_item(X): desbloqueia o item X

Para que a técnica de bloqueio binário possa ser usada, uma transação T deve obedecer às seguintes regras:

1. T deve emitir um lock_item(X) antes que qualquer read_item(X) ou write_item(X) seja executado;
2. T deve emitir um unlock_item(X) depois que todos os read_item(X) e write_item(X) tenham sido completados em T;
3. T não poderá emitir lock_item(X) se X estiver bloqueado por T;
4. T poderá emitir um unlock_item(X) apenas se tiver bloqueado X.

Algoritmo

lock_item(X):

B: se $LOCK(X) = 0$ então (* item desbloqueado *)

$LOCK(X) \leftarrow 1$ (* bloquear o item *)

senão início

 esperar até ($LOCK(X) = 0$ e o gerenciador de bloqueio despertar a transação);

 goto B;

fim;

unlock_item(X):

$LOCK(X) \leftarrow 0$; (* desbloquear o item *)

se alguma transação estiver esperando então

 despertar uma das transações em espera;

Bloqueio Binário

O Bloqueio Binário não pode ser feito em MySQL.

O MySQL não fornece diretamente um comando para "bloqueio binário". O que você pode fazer é configurar a replicação binária como mencionado nos passos anteriores para obter uma configuração semelhante a um "bloqueio binário", onde as alterações feitas em um servidor são replicadas para outro.

Bloqueio Múltiplo

Um esquema de bloqueio múltiplo (read/write ou compartilhado/exclusivo) permite que um item de dado seja acessado por mais de uma transação para leitura.

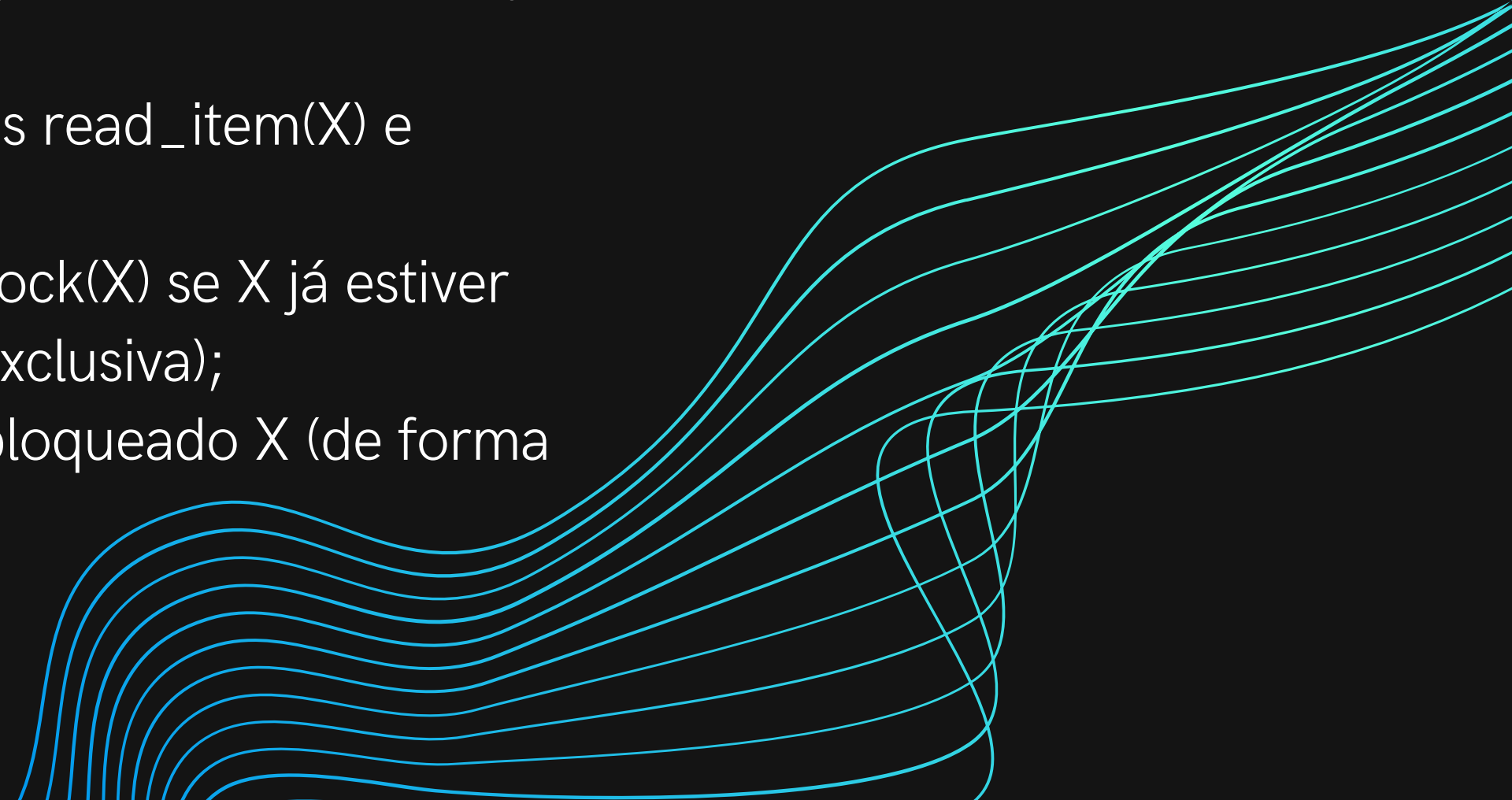
As operações necessárias são:

- `read_lock(X)`: bloqueia o item X para leitura, permitindo que outras transações leiam o item X (bloqueio compartilhado);
- `write_lock(X)`: bloqueia o item X para escrita, mantendo o bloqueio sobre o item X (bloqueio exclusivo);
- `unlock(X)`: desbloqueia o item X.

O MySQL não possui um conceito direto de "bloqueio múltiplo" no sentido de bloquear várias transações simultaneamente.

Bloqueio Múltiplo

Para que a técnica de bloqueio múltiplo possa ser usada, uma transação T deve obedecer às seguintes regras:

1. T deve emitir um `read_lock(X)` ou `write_lock(X)` antes que qualquer `read_item(X)` seja executado em T;
 2. T deve emitir um `write_lock(X)` antes que qualquer `write_item(X)` seja executado em T;
 3. T deve emitir um `unlock(X)` depois que todos os `read_item(X)` e `write_item(X)` tenham sido executados em T;
 4. T não emitirá nenhum `read_lock(X)` ou `write_lock(X)` se X já estiver bloqueado por T (de forma compartilhada ou exclusiva);
 5. T poderá emitir um `unlock(X)` apenas se tiver bloqueado X (de forma compartilhada ou exclusiva).
- 

Algoritmo read_lock

read_lock(X):

```
B: se LOCK(X) = "unlocked" então início (* item desbloqueado *)  
    LOCK(X) ← "read-locked"; (* bloquear o item p/ leitura *)  
    num_de_leituras(X) ← 1;  
fim  
senão se LOCK(X) = "read-locked" então (* bloqueado p/ leitura *)  
    num_de_leituras(X) ← num_de_leituras(X) + 1;  
senão início  
    esperar até (LOCK(X) = "unlocked" e o gerenciador de bloqueio  
                                     despertar a transação);  
    goto B;  
fim;
```

Algoritmo write_lock

write_lock(X):

```
B: se LOCK(X) = "unlocked" então  (* item desbloqueado *)  
    LOCK(X) ← "write-locked"  (* bloquear o item p/ escrita *)  
senão início  
    esperar até (LOCK(X) = "unlocked" e o gerenciador de bloqueio  
                                     desperta a transação);  
    goto B;  
fim;
```

Algoritmo unlock

unlock(X):

se LOCK(X) = "write_locked" então início (* bloqueado p/ escrita *)

 LOCK(X) ← "unlocked"; (* desbloquear o item *)

 despertar uma das transações em espera, se houver alguma;

fim

senão se LOCK(X) = "read-locked" então início (* bloqueado p/ leitura *)

 num_de_leituras(X) ← num_de_leituras - 1;

 se num_de_leituras = 0 então início

 LOCK(X) ← "unlocked"; (* desbloquear o item *)

 despertar uma das transações em espera, se houver alguma;

 fim;

fim;

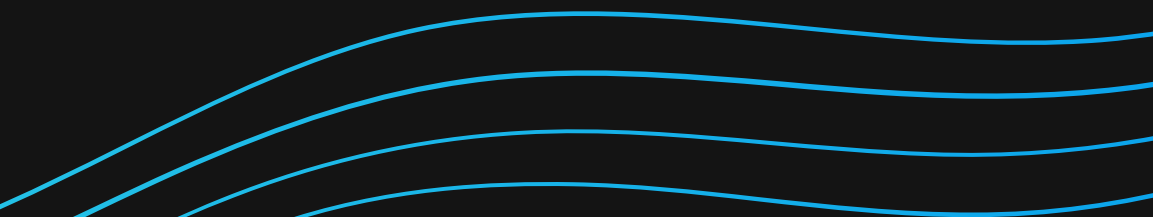
Bloqueio em Duas Fases

É uma técnica que combina o bloqueio com a utilização de duas fases: a fase de crescimento (ou aquisição de bloqueios) e a fase de encolhimento (ou liberação de bloqueios). Durante a fase de crescimento, as transações adquirem os bloqueios necessários para acessar e modificar os dados. Durante a fase de encolhimento, as transações liberam os bloqueios, permitindo que outras transações possam acessar os mesmos dados. Essa técnica garante que as transações sejam executadas de forma consistente e evita problemas como bloqueios em cascata. Para garantir escalonamentos serializáveis, as operações de bloqueio e desbloqueio nas transações devem seguir protocolos.



Bloqueio em Duas Fases

O conceito de "bloqueio em duas fases" geralmente está associado a sistemas distribuídos e protocolos de transações mais avançados, como o protocolo de duas fases do commit (2PC). No entanto, no MySQL, a implementação direta desse protocolo não é nativa.



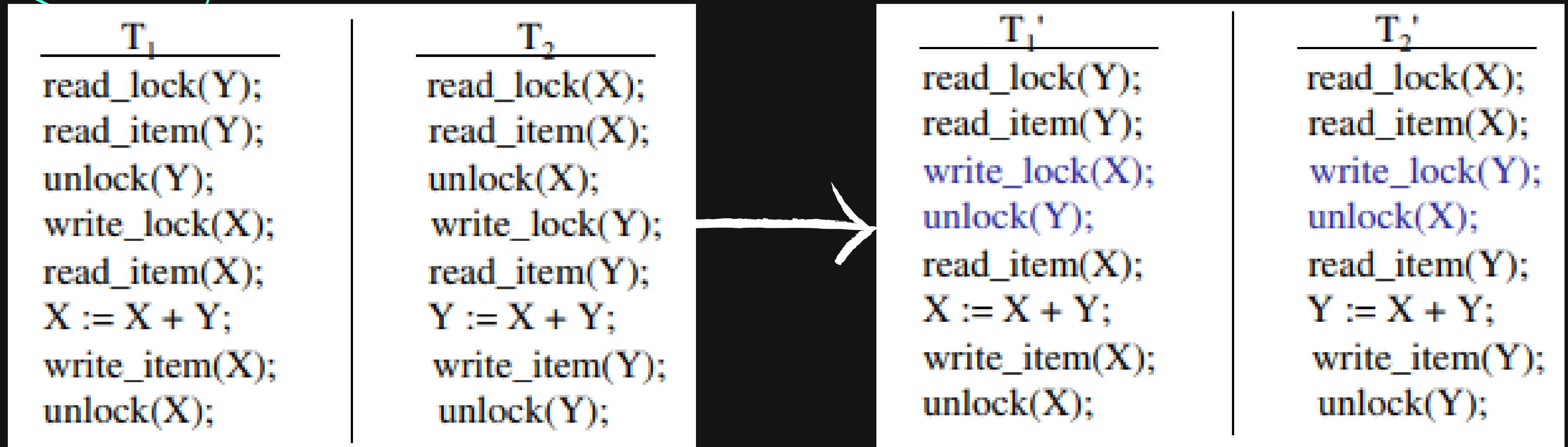
Protocolo

O protocolo mais usado é o protocolo de bloqueio em duas fases (Two-Phase Locking).

- Todas as operações de bloqueio (read_lock e write_lock) precedem a primeira operação de desbloqueio (unlock).
- As transações são divididas em duas fases:
 - a. expansão: quando são emitidos todos os bloqueios;
 - b. contração (encolhimento): quando os desbloqueios são emitidos e nenhum novo bloqueio pode ser emitido.



PARA SEGUIR O PROTOCOLO, AS TRANSAÇÕES
FORAM ALTERADAS PARA

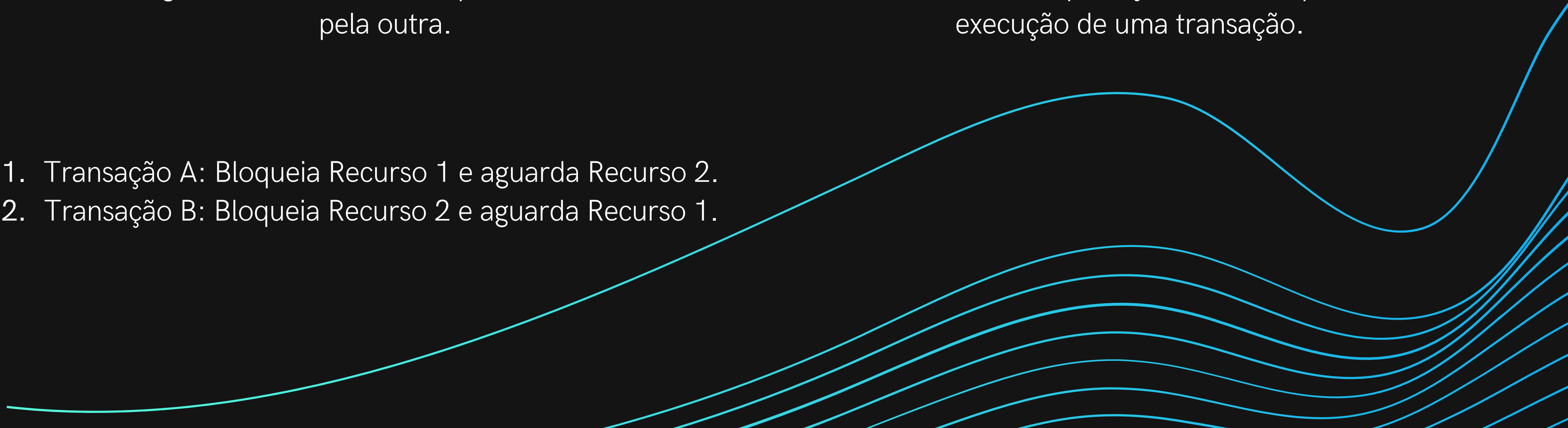


SE TODAS AS TRANSAÇÕES EM UM ESCALONAMENTO SEGUIREM O
PROTOCOLO DE BLOQUEIO EM DUAS FASES, O ESCALONAMENTO É
GARANTIDAMENTE SERIALIZÁVEL.

Deadlock

É uma situação em que duas ou mais transações estão bloqueadas permanentemente, incapazes de progredir, porque cada uma delas está aguardando um recurso que é mantido pela outra.


O deadlock é geralmente causado pela concorrência por recursos compartilhados, como linhas de uma tabela, páginas de dados ou qualquer outro recurso que seja necessário para a execução de uma transação.

1. Transação A: Bloqueia Recurso 1 e aguarda Recurso 2.
 2. Transação B: Bloqueia Recurso 2 e aguarda Recurso 1.
- 

Starvation

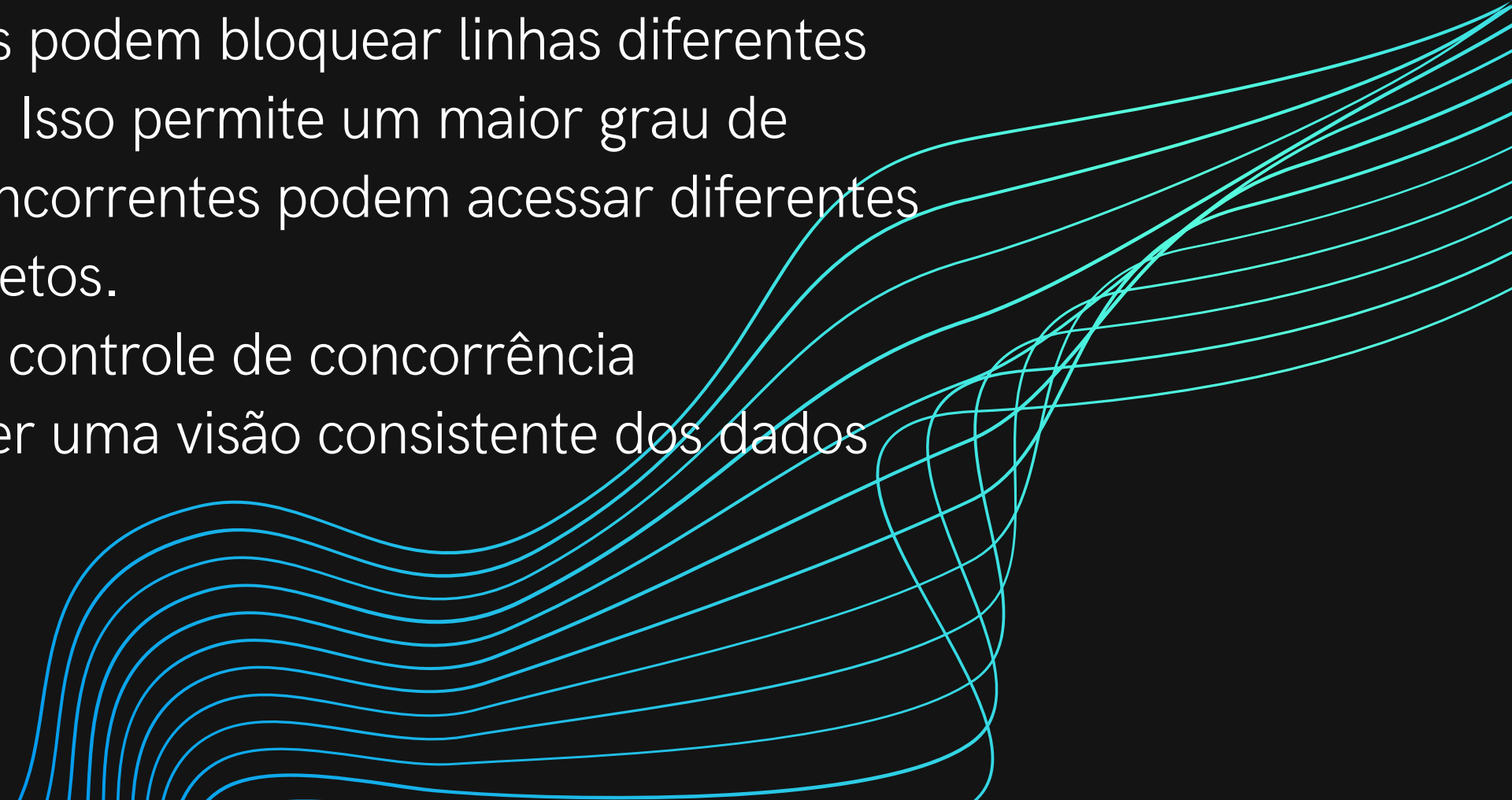
Ocorre quando uma transação é impedida de continuar sua execução indefinidamente, enquanto outras transações são executadas normalmente.

Na prática:

- O esquema de espera para itens bloqueados for injusto, priorizando algumas transações em relação a outras; Soluções:
 - Aumento de prioridade;
 - Fila de espera.
 - A mesma transação, no processo de seleção de vítima, for escolhida como vítima repetidamente. Solução:
 - Aumentar a prioridade de execução de transações que tenham sido abortadas inúmeras vezes.
- 

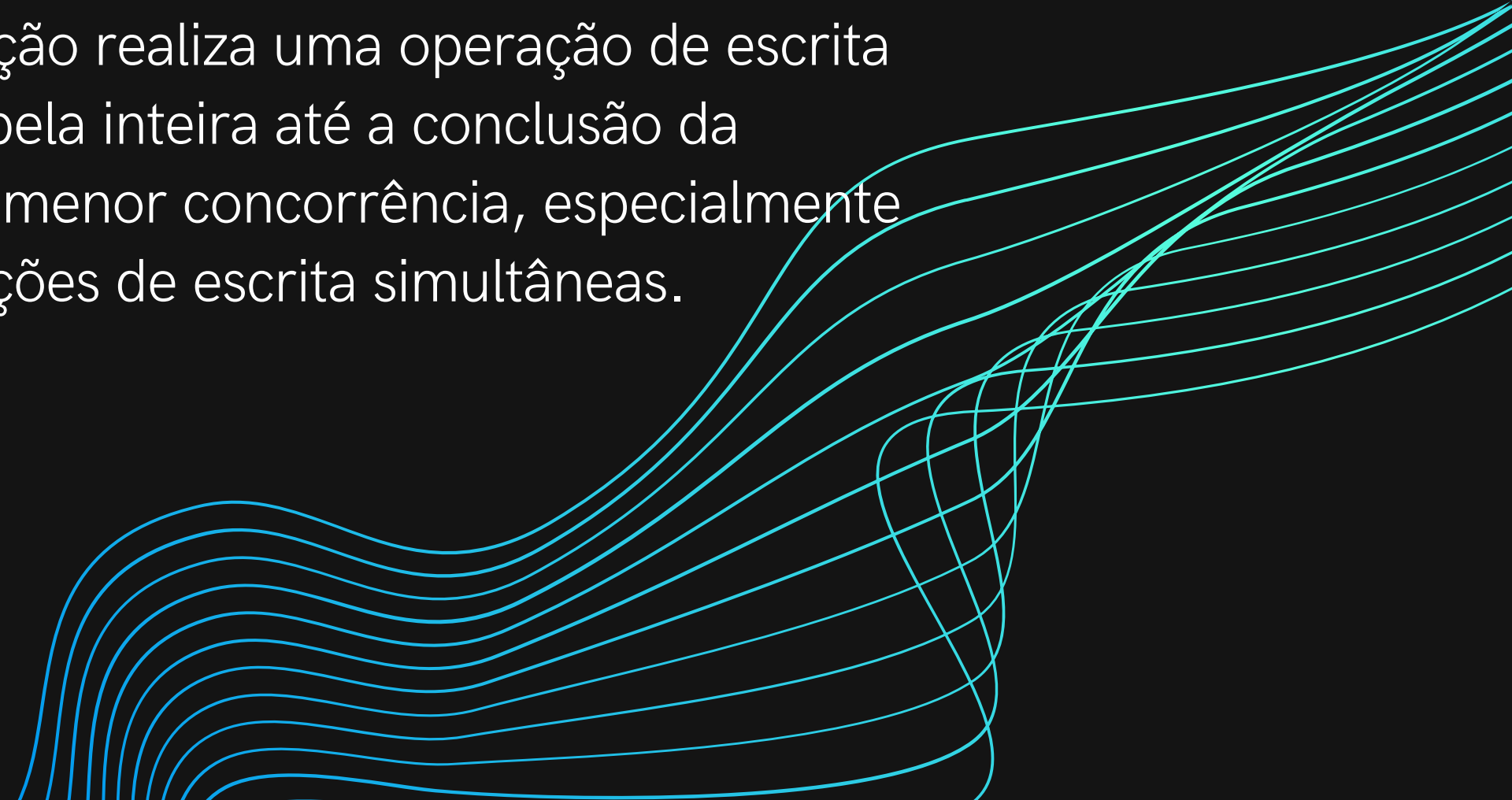
Qual Tipo de Bloqueio é o do MySQL?

InnoDB:

- O InnoDB é o mecanismo de armazenamento transacional padrão para o MySQL. Ele utiliza bloqueios a nível de linha (row-level locking) como parte de sua estratégia de controle de concorrência.
 - No InnoDB, diferentes transações podem bloquear linhas diferentes em uma tabela simultaneamente. Isso permite um maior grau de concorrência, pois transações concorrentes podem acessar diferentes partes da tabela sem conflitos diretos.
 - Além disso, o InnoDB emprega o controle de concorrência multiversão (MVCC) para fornecer uma visão consistente dos dados para diferentes transações.
- 

Qual Tipo de Bloqueio é o do MySQL?

MyISAM:

- O MyISAM é outro mecanismo de armazenamento no MySQL, mas ele utiliza bloqueio a nível de tabela (table-level locking) como estratégia de controle de concorrência.
 - No MyISAM, quando uma transação realiza uma operação de escrita em uma tabela, ela bloqueia a tabela inteira até a conclusão da transação. Isso pode resultar em menor concorrência, especialmente em ambientes com muitas operações de escrita simultâneas.
- 

FIM

